ISSN No. 0976-5697

Volume 2, No. 6, Nov-Dec 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# A Novel approach to Adaptively Secure Message Transmission in The Non-Erasure Model

Maged Hamada Ibrahim
Department of Electronics, Communications and Computers Engineering
Faculty of Engineering, Helwan University
1, Sherif st., Helwan, Cairo; Egypt
mhii72@hotmail.com

***Abstract:*** Secure message transmission can be solved perfectly assuming that the adversary controls less than one half of the paths connecting two nodes in a network while the rest of the paths are physically secured. Nevertheless, in many settings such assumption can be too strong to be realistic. While it may be reasonable to restrict the adversary full corruption, the adversary may have eavesdropping capabilities that extend over the whole network (beyond the corrupted nodes). In this case, we cannot run away of employing encryption techniques and hence, in addition to the fact that the security turns to be computational, adaptive adversaries come to play.
In this paper, we present a new Adaptively Secure Message Transmission (ASMT) that is based on any trapdoor function in the non-erasure model to allow any pair of nodes in a network to communicate in an adaptively secure manner assuming that the adversary observes (eavesdrops) the communications on all paths but actively corrupts a fraction of these paths. Our constructions are built using any available trapdoor function that is one-way secure.

***Keywords:*** Adaptive adversary, perfectly secure transmission, Simulation based security, Non-committing encryption, Non-erasure model.

## I. INTRODUCTION

Since the introduction of Secure Multiparty Computations (SMPC) in [1] through the well-known Yao's millionaires' problem, many contributions have appeared. The purpose of SMPC is to allow a set of parties (or nodes with honest majority) to compute a function using their private inputs in a private and robust way. An adversary that is able to corrupt (by altering inputs, deleting inputs, blocking transmission... etc.) a minority of the parties is not able to perform the computation of this function by her own or prevent the honest parties from performing the computation of this function correctly. Many applications appeared since the idea of SMPC introduced in [1], e.g. threshold signatures, electronic voting, electronic auctions, bargaining systems... etc.

The adversary is assumed to eavesdrop the channels connecting the communicating parties collecting all messages transferred among the parties, such an adversary may decide on which party to corrupt according to what she observes on the communicating channels and the information gathered so far from the already corrupted parties. Such adaptive behavior of an adversary motivated the need to device protocols that allows the incorporated parties to communicate in an adaptively secure way, that is, the adversary cannot distinguish between the real-life conversation among the parties and a fake simulated one.

### A. An overview of Existing Adversary Models:

There are several models of a corruptive adversary: *Stationary (non-mobile) adversary, Mobile adversary, Static adversary and Adaptive adversary*. In a stationary adversary, the adversary may attack a number of parties (minority), and this number is assumed not to exceed a certain value (the threshold) along the life time of the private inputs.

A mobile adversary [2, 3, 27] is able to jump from one party to the other (mobile virus attacks), collecting as much information as she can, she has the whole life-time of a secret to do so. Hence, the assumption that the adversary will not exceed a certain threshold no more holds. To withstand such type of an adversary, the parties must pro-actively renew their private inputs (cooperatively) through proactive security techniques and erase any previously shared information.

In a static adversary [4], the parties that the adversary is to corrupt are defined prior to the multiparty protocol execution, and remains unchanged during execution, that is, the adversary does not adapt her behavior during execution of the protocol whenever (for example) she finds that some party did not erase previous information after pro-actively renew her private inputs.

An adaptive adversary is the strongest known type of an adversary [4, 5]. This adversary is not only able to jump from one party to another, but she do that in a wise manner, according to her view of the communications among parties and her view of the computations of the already corrupted parties. Withstanding such type of adversaries is not an easy task especially in the existence of dishonest parties (non-erasing parties that are not trusted to erase their sensitive information).

Finally, there is another type of an adversary that cannot corrupt a party, yet, she has a coercive power that allows her to coerce a party to do as she wishes. This type of an adversary is known as a coercive adversary [6, 7, 28, 29]. The notion of *deniable encryption* deals with this type of an adversary in the sense that it allows a party to open any plaintext message that when verified gives the same ciphertext observed by this adversary.

### B. The Non-Erasure Model:

The main difficulty in developing multiparty protocols that can be proven secure against adaptive attacks is the fact that most cryptosystems bind a party to the plaintext message, even though the plaintext itself may be hard to calculate from the ciphertext alone. Another difficulty is that, in many cases even if a party is willing to erase her sensitive information, she cannot do so, for example, a party in possession of her private key (corresponding to a published public key) is not able to erase her secret key (although she is willing to do that) since in this case she is not able to decrypt the outstanding ciphertexts. An adversary can simply observe a ciphertext $C = E(m, r)$ over a public line (for some message *m* and associated random coins *r*). The adversary is expecting (when later corrupts the sending party) to see the random coins *r* used in association with *m* to produce *C*. Also, the adversary is expecting to see the secret key $sk$ on the receiver's side when corrupting the receiver. Notice that, in the erasure model (when both parties are willing to erase their data), the sender is able to erase his random coins immediately after encryption, but unfortunately, the receiver fails to obey such erasure requirement of her $sk$. This motivates the need to design cryptographic primitives to be adaptively secure in the *non-erasure* model, i.e. a cryptographic primitive that does not require any party to erase nothing.

### *C. Simulation: The Ideal Functionality:*

In computational security, zero-knowledge approaches are a common standard for demonstrating interactive security [14, 15, 16, 17, 18]. Typically, one must find a simulator that presents a convincing but faked conversation without having access to the private information that normally may play a role in generating the actual conversation. If the fake conversation is indistinguishable from a real one, then we may infer that the real one leaks no "knowledge" about the sensitive private information.

The security of a protocol should be satisfied with the following: the adversary (also spoken of as the environment) $Z$'s view about the protocol $\pi$ in the real-life model can be simulated by a dummy adversary $S$ that runs in the ideal model. In the real-life model, the parties execute the protocol $\pi$. The adversary $Z$ is able to see the internal data of corrupted parties and the protocol messages of $\pi$. In the ideal model, we still have $Z$, but $\pi$ is replaced by an ideal functionality $F$ (think of it as a trusted third party).

The incorporated $n$ parties only hand their private inputs $(x_1, \ldots, x_n)$ to $F$, then $F$ performs the computations and outputs $(y_1, \ldots, y_n)$ where $y_i$ is party *i*'s output, yet there could be a single output $y$ common to all parties.

In the ideal model, the protocol $\pi$ does not exist no more, neither do the parties, they are replaced with $F$, and a simulator $S$ (a.k.a an interface and may be regarded as an adversary) is presented in order to supply fake internal data and protocol messages in the ideal model to $Z$. A protocol $\pi$ is said to be adaptively secure if the environment $Z$ cannot distinguish between a real-life execution of $\pi$ and a simulated one.

### *D. Adaptively Secure Encryption:*

Adaptively secure encryption represents the tool (plug and play) to achieve adaptive security in multiparty protocols. This tool is a.k.a non-committing encryption. However, the term non-committing is misleading because such encryption is indeed committing as notified by Beaver in [19]. This encryption is committing in the sense that an honest sender cannot later pretend that an alternate message was sent. That is, these cryptosystems are non-committing in the existence of the simulator (the ideal world, not the real world). Here, we distinguish between deniable encryption [6] and non-committing (adaptively secure) encryption. Deniable encryption is a true non-committing encryption which faces a type of an adversary known as a coercive adversary. This adversary is weaker than a corruptive adversary in the sense that, she cannot corrupt a party, yet, she has some power that allows her to coerce this party to do as she wishes. In deniable encryption, a sender can generate a ciphertext that appears as an encryption of two or more different messages, whereas, in non-committing encryption, the ciphertext that could be opened as an encryption of any message is generated by the simulator (in the ideal world).

## II. RELATED WORK

Among the different contributions to come up with an efficient non-committing encryption scheme, the scheme of Beaver [19] represents the most efficient scheme that provides a tool for adaptive security in the non-erasure model. His scheme is an extension to the DH key exchange to allow one bit transmission from a sender to a receiver in an adaptive manner. In this scheme, the sender holds a random bit $c$ while the receiver holds a random bit $d$. Beaver's scheme allows both **S** and **R** to test whether $c = d$. If equality holds, a one bit message $m$ is transmitted as $m \oplus c$, else, the attempt fails. The scheme succeeds in transmitting a one bit message with probability 1/2, hence, it suffers from possible fail attempts which is not suitable for many real-life applications (e.g. realtime communications). The scheme is a three-pass (three rounds) scheme between **S** and **R**, each pass is in the order of $\mathrm{O}(\lambda)$ bits for a security parameter $\lambda$.

Another contribution is proposed in [20] as so called simulatable public-key encryption, which is essentially Beaver's idea and is weaker than Beaver's scheme. Notice that Beaver's scheme requires no erasure at all and hence it is secure in the non-erasure model, while the work in [20] assumes that the sender erases his random coins after encryption. The technique is that the receiver picks two public keys, $P_0$ and $P_1$, one of which he knows the corresponding secret key $sk_d$, $d \in_R \{0,1\}$, and the other public key is picked obliviously. The sender picks two random messages $M_0$ and $M_1$ from the message space, he picks $c \in_R \{0,1\}$ and encrypts $M_c$ using $P_c$ to produce a ciphertext $C_c$ while $C_{1-c}$ is sampled at random from the ciphertext space. He sends $M_0, M_1, C_0, C_1$ to the receiver. The receiver decrypts $C_d$ using $sk_d$ and finds out whether the resulting plaintext matches $M_d$. If a match exists then $c = d$, else, $c \neq d$. The sender then uses $c$ to encrypt a one bit message $m$. Notice that the scheme of [20] is a stronger test of equality in the erasure model, in which, there are no fail attempts except with negligible probability.

The work in [21] showed that, in the programmable random oracle model (PROM), when all parties have access to a public programmable random oracle, it is easy to construct a non-committing encryption scheme using the IND-CPA public key encryption scheme $C = \langle f(r), m \oplus H(r) \rangle$ (introduced in [22]), where $r$ is the random coins picked by the sender, however, when the random oracle is replaced with a practical hash function (locally instantiated), the simulation fails since in this case the simulator cannot program the RO no more.

The scheme in [23] requires that the sender erases his random coins after encryption and hence it is secure in the weaker model of adaptive security (the erasure model). The idea is to allow the simulator $S$ to give the environment $Z$ a garbage ciphertext (which is a function of the receiver's secret key). Later, when $S$ knows the plaintext message $m$, he is able to fabricate a fake secret key for $Z$ that decrypts this garbage ciphertext to $m$. The scheme requires that the message space is small since the receiver recovers $m$ using exhaustive search. Their scheme is based on the public key encryption scheme of [24] which is an extension of the IND-CPA El-Gamal encryption [25] to satisfy the CCA-1 security model. They devised another scheme based on the public key encryption introduced in [26] that allows decrypting exponentially large messages. Their schemes are called receiver non-committing (RNC), since the sender is still committed to the ciphertext unless he erases his random coins. However, in RNC encryption, when $S$ opens more than one fake secret key to $Z$ for more than one garbage ciphertext, $Z$ is able to try these keys on different previous garbage ciphertexts and hence is able to distinguish the real-life protocol from the simulated one. To overcome this difficulty, the authors applied forward secure encryption techniques to their RNC (AFS-RNC) under the limitation that, only one ciphertext could be transmitted per time period. To transmit $\ell$ ciphertexts per time period, the receiver must publish $\ell$ different public keys and the sender performs $\ell$ encryptions to produce one ciphertext. In addition to the fact that their scheme satisfies a weaker notion of adaptive security, the scheme is very complex requiring a lot of work on the sender's and the receiver's side.

In perfectly secure message transmission (PSMT) first introduced in [8], and improved in subsequent contributions (e.g. [9,10,11,12,30,31]), a sender **S** and a receiver **R** are connected by $n = 3t + 1$ channels with at most $t$ channels are corrupted by the adversary, while the remaining $n - t$ channels are beyond the reach of the adversary (physically secured). Under these assumptions (in one round of communication) **S** is able to transmit a message $m$ to **R** in a perfectly secure way using polynomial sharing. Finally, **R** decodes for the message using the well-known BerleKamp-Welch decoder [13]. In three rounds of communication (assuming that **S** is the party that always start the communication) connectivity could be improved to $n = 2t + 1$. Such perfect secure transmission is indeed adaptively secure, since it could be easily shown that a simulator will always be able to fake a conversation as long as at least $n - t$ channels are beyond the reach of the adversary. Under the assumption that physically secure channels exist between every pair of nodes in the network, the schemes introduced in this paper are not useful. Yet, the assumption that some channels are physically secured is impractical in many applications (e.g. the internet) and hence, standard encryption techniques are employed to protect from adversary with eavesdropping capacities that extend to the whole network. In this case, privacy is preserved (in the cryptographic sense) and correctness is achieved by assuming that the adversary corrupts a fraction of the network paths. However, in addition to the fact that the system becomes cryptographically secure, employing standard encryption techniques gives rise to adaptive vulnerabilities. An adaptive adversary viewing all communications between **S** and **R** is capable of adapting here behavior according to what she observes on the channels.

From the above discussion, we conclude that, if physically secured channels are not available then realizing PSMT is impossible. One cannot run away from employing cryptographic techniques and thus, adaptive adversaries come to play.

## III. MOTIVATIONS AND CONTRIBUTIONS

### A. *Motivations:*

Adaptive security is an important notion in secure multiparty protocols to withstand the strongest type of an adversary (adaptive adversary). To realize adaptively secure protocols, incorporated parties are required to communicate in an adaptively secure way. One way to realize adaptive security is via PSMT; however, PSMT relies on the assumption that physically secure channels are available which is impractical for many applications. To get rid of physically secure channels, encryption techniques are employed and hence, transmission is vulnerable to adaptive attacks. Among the schemes proposed so far to withstand adaptive adversaries, Beaver's scheme [19] is the most efficient and is the only scheme in the non-erasure model, yet, it requires extensive amount of communications and computations, also, it relies on the idea of equality tests and hence suffers from the possibility of failure, a one bit transmission succeeds with probability 1/2. All other schemes proposed so far are in the erasure model requiring the parties to erase some sensitive information during computation and hence are secure in the weakest model of adaptive security (the erasure model).

### B. *Contributions:*

Under the assumption that the adversary's eavesdropping capabilities extend to the whole network but corrupt a fraction of the available paths, we propose an adaptively secure message transmission (ASMT) protocol allowing $n$ parties (or nodes) to communicate in an adaptively secure manner. The protocol is adaptively secure in the non-erasure model, i.e. no party is required to erase any internal data. It could be realized using any trapdoor function that is one-way secure. It is simulatable and correct. We first build a one pass sender non-committing encryption (SNCE) from any trapdoor function, then from our SNCE we build a three-pass receiver non-committing encryption scheme RNCE. Given the assumed network description, using our SNCE and RNCE, we realize a wiring functionality that when combined with a PSMT protocol gives rise to an efficient ASMT protocol construction in the non-erasure model.

We show that our proposed schemes are simulatable in the ideal world and hence satisfy the security requirements to face an adaptive adversary.

## IV. OUR DEVELOPED TOOLS

In this section we develop and analyze the basic tools used to construct our ASMT protocol. We first develop a sender non-committing encryption scheme, and then we use this scheme to develop a receiver non-committing encryption scheme.

### A. *Sender Non-Committing Encryption:*

**Definition.** A protocol $\pi$ with sender **S** and receiver **R**, and with security parameter $\kappa$ is a sender non-committing encryption (SNCE) scheme for a message bit $m$ if:

a. ***Correctness*:** The probability that **R**'s output is different from **S**'s input is negligible.
b. ***Security*:** The communication for transmitting $m$ is computationally indistinguishable from the communication for transmitting $\overline{m}$.
c. ***Non-commission*:** There is a simulator that when attacking the interaction between **S** and **R** in the secure channel model, can provide the environment $Z$ (that may only corrupt **S**) with a fake view that is indistinguishable from the real one.

**Real world scheme.** The public encryption key $pk$ allows the sender **S** to generate uniformly at random a member of a translucent set $T_\lambda$ [6]. A translucent set was implemented in [6] using hardcore bits. We give a simpler method to compute an element of a translucent set c as follows: Let $(f, f^{-1})$ be a one way trapdoor permutation, pick $R \in_r D_f$, compute the tuple $\langle H(R), f(R)\rangle$ as an element of $T_\lambda$ where $H:\{0,1\}^* \to \{0,1\}^k$ is a suitable hash function. The private decryption key $sk$ allows the receiver to distinguish elements in the translucent set $T_\lambda$ from random elements in $\{0,1\}^\lambda$. The SNCE scheme is described next.

**Encryption: SNCE** $(pk, m)$. The sender encrypts a one-bit message $m$ as follows:

a. If $m=1$, computes a random element from $T_\lambda$ as the ciphertext $\psi$. Else, picks a random element from $\{0,1\}^\lambda$ as the ciphertext $\psi$.
b. Sends $\psi$ to the receiver.

**Decryption: SNCD** $(sk, \psi)$ .} On the reception of $\psi$, the receiver (using his trapdoor secret key $sk$ ) checks whether $\psi$ is in $\psi$ and decides on $m$.

**Correctness.** The receiver decides on the correct $m$ except with negligible probability $2^{-\kappa}$.

Semantic security is straightforward. We prove that our SNCE does not commit the sender to the encrypted bit and hence it is adaptively secure against sender corruption.

**Lemma 1: Adaptive security.** Given that the underlying trapdoor function is one-way secure, the above scheme is adaptively secure against sender corruption.

**Proof.** We prove this lemma via simulation: As usual, in the ideal model, there is a simulator $S$ that when attacking the interaction between **S** and **R** in the secure channel model, can provide the environment $Z$ with a fake view that is indistinguishable from the real one. We assume that the receiver's public parameters are known to $S$ and $Z$. We assume a sender corruption that is, the receiver is beyond corruption. The simulation (without corruption) runs as follows:

a. Picks a random element in $T_\lambda$ as the fake ciphertext $\psi$.
b. Reports $\psi$ as the transmission **S→R**.

Now, when the sender is corrupted and $S$ knows the message bit $m$, $S$ can show to $Z$ that the fake ciphertext $\psi$ (that he is committed to before knowing $m$ ) is an encryption of $m$. Simply, if $m=1$, he honestly opens the element in $T_\lambda$. If $m=0$, he claims that $\psi$ is picked at random from $T_\lambda$. Using $pk$, $Z$ is able to verify the correctness of $S$'s claim (i.e. that the opened values (in case $m=1$) encrypts to $\psi$, but since the receiver **R** is beyond corruption, $Z$ has no access to the trapdoor secret key $sk$ and hence, cannot detect that $S$ is cheating (in case $m=0$) when claiming that $\psi \in_r \{0,1\}^\lambda$, or else, she is able to break the one-wayness of the trapdoor function.

## *B. Receiver Non-Committing Encryption:*

The definition of RNCE is analogous to the definition of SNCE. We convert our SNCE to a RNCE. The scheme is essentially a two-pass scheme but since (in practice) the sender always starts the communication, we consider the scheme as a three-pass scheme. Let $m$ be the bit to be encrypted and transmitted from **S** to **R**. **R** chooses a random bit $r$ and invokes scheme SNCE to encrypt and send $r$ to **S** (as if **S** and **R** have exchanged places). **S** replies with $c = r \oplus m$ in the clear. Notice that, in this case $sk$ is held by **S** and hence, **S** is assumed beyond corruption. In more details, the scheme is as follows:

**Encryption: RNCE** $(pk, m)$. For **S** to encrypt a one-bit message $m$ to **R**:

a. **R→S**:
   i. Picks $r \in_r \{0,1\}$.
   ii. If $r=1$, picks and sends a random element $y$ from $T_\lambda$. Else, picks and sends a random element $y$ from $\{0,1\}^\lambda$.
b. **S→R**:
   i. Finds whether $y$ is in $T_\lambda$ and decides on $r$.
   ii. Computes and sends the ciphertext $\psi = r \oplus m$.

**Decryption: RNCD** $(r, \psi)$. **R** computes $m = r \oplus \psi$.

**Lemma 2: Adaptive security.** Given that the underlying trapdoor function is one-way secure, the above scheme is adaptively secure against Receiver corruption.}\end{lemma}

**Proof.** We prove that our scheme does not commit the receiver to the received message *m* and hence it is a receiver non-committing encryption. We assume that the sender is beyond corruption. The simulation without corruption is as follows:

a. $S \to Z$ Simulating transmission **R→S**. Picks and reports a random element $\tilde{y}$ from $T_\lambda$.
b. $S \to Z$ Simulating transmission **S→R**. Picks and reports $\tilde{\psi} \in_r \{0,1\}$.

We then define the three possible stages of corruptions:

i. Stage 0: **R** corruption before transmission.
ii. Stage 1: **R** corruption after transmission **R→S**.
iii. Stage 2: **R** corruption after transmission **S→R**.

Next we patch **R**'s view in each stage. In stage 0, there is nothing to patch. In stage 1, $S$ picks $\tilde{r} \in_r \{0,1\}$, in case $\tilde{r}=1$, patches **R**'s view with ($\tilde{r}$, $\tilde{y}$, $\tilde{x} \in T_\lambda$) and reports it to $Z$, in case $\tilde{r}=0$ he patches **R**'s view with ($\tilde{r}$, $\tilde{y}$) and reports it to $Z$. In stage 2, $S$ is already committed to $\tilde{\psi}$, he now knows $m$ from corrupted **R**, he computes

$\tilde{r} = \tilde{\psi} \oplus m$, in case $\tilde{r} = 1$, patches **R**'s view with ($\tilde{r}$, $\tilde{y}$, $\tilde{x} \in T_\lambda$) and reports it to $Z$, in case $r = 0$ he patches **R**'s view with ($\tilde{r}$, $\tilde{y}$) and reports it to $Z$.

## V. DESCRIPTION OF THE ASMT PROTOCOL

We first describe our protocol in the existence of an adaptive-but-passive adversary which is an adversary that is able to behave adaptively but is not able to corrupt any of the intermediate nodes between the sender and the receiver. Then, we describe and simulate the protocol in the existence of a fully corruptive adversary.

### *A. The Protocol Against Adaptive-but-Passive Adversary:*

**A single-path protocol.** We first describe a protocol $\pi'_{ASMT}$ which is a reduced version of the protocol $\pi_{ASMT}$ in the existence of only one path between **S** and the **R** with only one intermediate node **T** that is assumed beyond corruption and which is able to learn the message $m$ sent from **S** to **R**. Since **T** is not corrupted her trapdoor inverse $f^{-1}_{pk_T}$ is beyond the reach of the adversary. For **S** to send a one-bit message $m$ in an adaptively secure way:

a. Using **T**'s public key $pk_T$, both **S** and **T** run the **SNCE** $(pk_T, m)$ to transmit $m$ to **T**. At the end, **T** knows $m$.

b. Using **T**'s public key $pk_T$, both **T** and **R** run the RNCE $(pk_T, m)$ to transmit $m$ to **R**. At the end, **R** knows $m$.

Protocol $\pi'_{ASMT}$ is a secure sender and receiver non-committing encryption scheme that realizes the ideal functionality $F_{ASMT}$ against an adaptive-but-passive adversary.

**The multi-path protocol.** We consider the protocol $\pi_{ASMT}$ which is now an extension of the protocol $\pi'_{ASMT}$. In $\pi_{ASMT}$, **S** and **R** are connected via $k$ paths $p_1, \cdots, p_k$. Since it is assumed that when any node on a path is corrupted, the whole path is down, for simplicity and wlog, we assume that there is only one intermediate node on each path. The $k$ nodes are assumed honest-but-non-erasing. Let $N = \{N_1, \cdots, N_k\}$ be the set of the $k$ nodes.

Again, since the parties are honest, one may consider the adversary as adaptive-but-passive. The adversary is assumed to view all data on all paths and that among the $k$ nodes, there exists at least one node beyond the reach of the adversary. To transmit a one bit message from **S** to **R**, **S** picks $k$ bits at random $s_1, ..., s_k$ such that $m = \oplus_{i=1}^{k} s_i$. The protocol $\pi_{ASMT}$ proceeds as follows:

a. **S** invokes the **SNCE** $(pk_i, s_i)$ to transmit $s_i$ to node $N_i$ using $N_i$ 's public key $pk_i$ $\forall i = 1, \cdots, k$.

b. $\forall i = 1, \cdots, k$, each node $N_i$ invokes the RNCE $(pk_i, s_i)$ using her public function $pk_i$ to transmit $s_i$ to **R**.

c. **R** collects the $k$ bits $s_1, ..., s_k$ and reconstructs the message $m$.

By considering each bit $s_i$ as a separate message, it is easily seen that protocol $\pi_{ASMT}$ is a sender-and-receiver adaptively secure protocol for one bit encryption that realizes $F_{ASMT}$ against an adaptive-but-passive environment assuming at least one node in $N$ beyond the reach of the adversary.

### *B. The Protocol Against Fully Corruptive Adversary:*

Let $P = p_1, \cdots, p_k$ be a set of $k \geq 3t + 1$ available paths between the sender and the receiver and let $N = \{N_1, \cdots, N_k\}$ be a set of $k$ nodes. Wlog, assume only one intermediate node $N_\ell$ on each path $p_\ell$. There are at most $t$ corrupted intermediate nodes for a threshold $t$. Assume that the adversary eavesdrops all communication channels. Let $N_s$ and $N_r$ be the sender and the receiver respectively. To transmit a $b$-bit message $m$ from $N_s$ to $N_r$, $N_s$ constructs a $t$-degree polynomial $g(x)$ over a prime field $Z_p$ for a prime $p > \max(k, 2^b)$. He sets $g(0) = m$. The protocol $\pi_{ASMT}$ is as follows:

a. $N_s$ executes the **SNCE** $(pk_i, s_i)$ to transmit a share $s_i = g(i)$ to each node $N_i$ using $N_i$ 's public key $pk_i$, $\forall i = 1, \cdots, k$.

b. $\forall i = 1, \cdots, k$, each party $N_i$ executes the **RNCE** $(pk_i, s_i)$ using her public key $pk_i$ to transmit $s_i$ to $N_r$.

c. $N_r$ collects the $k$ shares $s_1, ..., s_k$ and interpolates for $m$ using the Berlekamp-Welch decoding scheme.

**Simulation.** Given that the SNCE and the RNCE are already proven adaptively secure. The simulation without corruption is as follows:

a. $S \to Z$ Simulating transmission $N_s \to N$. Picks and reports $k$ random elements $s_1, ..., s_k$ in $Z_p$ as shares of a random $t$-degree polynomial $g(x)$ encrypted under the simulated version of **SNCE** $(pk_i, \tilde{s}_i)$.

b. $S \to Z$ Simulating transmission $N_r \to N$. Picks and reports $k$ random elements $\tilde{r}_1, \cdots, \tilde{r}_k$ in $Z_p$ encrypted under the simulated version of **SNCE** $(pk_i, \tilde{r}_i)$.

c. $S \to Z$ Simulating transmission $N \to N_r$. Picks and reports $k$ random elements $(\tilde{\psi}_1, \cdots, \psi_k)$ in $Z_p$.

Next, we define the following corruption stages:

i. Stage 0: Before transmission.

ii. Stage 1: After transmission $N_s \to N$.

iii. Stage 2: After transmission $N_r \to N$.

iv. Stage 3: After transmission $N \to N_r$.

Let $N' \to N$ be a subset of any $t$ parties. Wlog let $N' = \{N_1, \cdots, N_t\}$. We further divide each stage into three sub-stages according to which node $Z$ chooses to corrupt first.

a. **Stage-0-** $N_s$ **:** $S$ knows $m$ and reports $(\tilde{s}_1, \cdots, \tilde{s}_k)$ consistent with $m$.

b. **Stage-0-** $N_r$. There is nothing to report.

c. Stage-0- $N'$. There is nothing to report.

d. Stage-1- $N_s$. $S$ knows $m$, reports $(\tilde{s}_1, \cdots, \tilde{s}_k)$ consistent with $m$ and patches $N'$ view as $(\tilde{s}_{i_1}, \cdots, \tilde{s}_{i_t})$.

e. Stage-1- $N_r$. There is nothing to report.

f. Stage-1- $N'$. Reports random $(\tilde{s}_1, \cdots, \tilde{s}_t)$ in $Z_p$ and patches $N_s$ view as $(\tilde{s}_1, \cdots, \tilde{s}_k)$ where $(\tilde{s}_{t+1}, \cdots, \tilde{s}_k)$ are decided when $m$ is known from corrupted $N_s$.

g. Stage-2- $N_s$. $S$ knows $m$ and reports $(\tilde{s}_1, \cdots, \tilde{s}_k)$ consistent with $m$. Patches $N'$ view as $(\tilde{s}_1, \cdots, \tilde{s}_t, \tilde{r}_1, \cdots, \tilde{r}_t)$ and patches $N_r$ view as $(\tilde{r}_1, \cdots, \tilde{r}_k)$ where $r_i \in_r Z_p$.

h. Stage-2- $N_r$. Reports $(\tilde{r}_1, \cdots, \tilde{r}_k)$, patches $N'$ view as $(\tilde{s}_1, \cdots, \tilde{s}_t, \tilde{r}_1, \cdots, \tilde{r}_t)$ and patches $N_s$ view as $(\tilde{s}_1, \cdots, \tilde{s}_k)$ where $(\tilde{s}_{t+1}, \cdots, \tilde{s}_k)$ are decided when $m$ is known from corrupted $N_s$.

i. Stage-2- $N'$ .} Reports $(\tilde{s}_1, \cdots, \tilde{s}_t, \tilde{r}_1, \cdots, \tilde{r}_t)$, patches $N_s$ view as $(\tilde{s}_1, \cdots, \tilde{s}_k)$ where $(\tilde{s}_{t+1}, \cdots, \tilde{s}_k)$ are decided when $m$ is known from corrupted $N_s$. Patches $N_r$ view as $(\tilde{r}_1, \cdots, \tilde{r}_k)$ where $r_i \in_r Z_p$.

j. Stage-3- $N_s$ .} $S$ knows $m$ and reports $(\tilde{s}_1, \cdots, \tilde{s}_k)$ consistent with $m$. He is committed to $(\tilde{\psi}_1, \cdots, \tilde{\psi}_k)$. Patches $N'$ view as $(\tilde{s}_1, \cdots, \tilde{s}_t, \tilde{r}_1, \cdots, \tilde{r}_t)$ and patches $N_r$ view as $(\tilde{r}_1, \cdots, \tilde{r}_k)$ where $(\tilde{r}_i = \tilde{s}_i \oplus \tilde{\psi}_i)$ for *i*=1,…,*k*.

k. Stage-3- $N_r$. $S$ knows $m$ from corrupted $N_r$. He is committed to $(\tilde{\psi}_1, \cdots, \tilde{\psi}_k)$. He Picks $(\tilde{s}_1, \cdots, \tilde{s}_k)$ consistent with $m$ and computes $(\tilde{r}_i = \tilde{s}_i \oplus \tilde{\psi}_i)$ for *i*=1,…,*k*. Reports $(\tilde{s}_1, \cdots, \tilde{s}_k, \tilde{r}_1, \cdots, \tilde{r}_k)$, patches $N'$ view as $(\tilde{s}_{i_1}, \cdots, \tilde{s}_{i_t}, \tilde{r}_{i_1}, \cdots, \tilde{r}_{i_k})$ and patches $N_s$ view as $(\tilde{s}_1, \cdots, \tilde{s}_k)$.

l. Stage-3- $N'$. Reports $(\tilde{s}_{i_1}, \cdots, \tilde{s}_{i_t}, \tilde{r}_{i_1}, \cdots, \tilde{r}_{i_k})$, patches $N_s$ view as $(\tilde{s}_1, \cdots, \tilde{s}_k)$ and $N_r$ view as $(\tilde{r}_1, \cdots, \tilde{r}_k)$ where $(\tilde{s}_{t+1}, \cdots, \tilde{s}_k)$ and $(\tilde{r}_{t+1}, \cdots, \tilde{r}_k)$ are decided among corrupting either $N_s$ or $N_r$ as before.

Since at most $t$ intermediate nodes are corrupted by $Z$, then the SNCE and the RNCE allow the simulator $S$ (without knowing $m$) to come up with $t$ random elements in $Z$ as fake $t$ shares of a polynomial $\tilde{g}(x)$. These $t$ random shares do fix the polynomial and hence $S$ is not committed to a particular $m$. When $m$ is later known to $S$, he is still able to come up with an additional share that fixes $\tilde{g}(x)$ to be consistent with $m$.

## VI. COMPARISONS AND EVALUATION

### A. *Communications Overheads:*

For a security parameter $\lambda = s + \kappa$ where $s$ describes the domain of the trapdoor function, it requires $N_s$ to receive $k$ public keys from its neighbors, each of length $s$ and to send $\lg p$ bits encrypted using the non-committing encryption on each of the $k$ paths, collecting $ks + k\lambda \lg p$ bits.

On the $N_r$ side, it requires $ks$ bits to receive the neighbors' public keys, $k \lg p$ bits encrypted using the non-committing encryption and $k \lg p$ bits back to $N_r$. Finally, assuming $z$ intermediate nodes on each path communicating using a committing encryption scheme of security parameter $s$, we have $zs$ for the public key and $zs$ for the ciphertext (assuming $s > \lg p$) on each path. This totals $2ks + 2k\lambda \lg p + k \lg p + 2zsk$. If $m$ is of $\lg p$ bits then the communications overheads is $\Lambda \approx 2ks \lg p + 2k\lambda l + 2zsk \lg p$.

Comparing our scheme (which is fully adaptive in the non-erasure model) to any of the previously proposed schemes that allow parties to erase any of their internal data (erasure-model) is completely unfair since constructions in the erasure model is much easier than constructions in the non-erasure model. Among the schemes proposed so far, Beaver's bit equality test scheme (BET) [19] is fully adaptive scheme in the non-erasure model and represents the most competitive scheme to ours. Hence, in the following, we compare our ASMT to the BET scheme.

### *B. Possbiility of Failure:*

One major advantage of our scheme over the BET scheme is that our scheme is a one attempt scheme with no possibility of failure (except with negligible probability) whereas, the BET scheme fails with probability 1/2.

### *C. General Assumptions:*

The BET scheme is an extension to the DH key exchange and hence the security is based on the DH problem, on the other hand, our constructions are general, in the sense that we do not rely on any specific assumptions about the trapdoor door function in use other than its one-wayness.

Since the BET scheme may obviously replace our SNCE and RNCE, in our ASMT protocol we replace the SNCE and the RNCE with the BET scheme and concretely compare the computation and communication complexities in both cases on the sender's side and on the receiver's size.

### *D. Computation Complexity:*

On the sender's side: in BET scheme, the sender performs at least two exponentiations (independent of the message bit *m* to be sent) while in our SNCE, in case *m*=0, the sender performs nothing but picking random elements and *performs no computations at all* while in case *m*=1 only one trapdoor function is computed and $\lambda$ hardcore predicates. So, on the sender's side, the computation complexity (per bit) is efficiently reduced. The same computation complexity reduction also holds on the receiver's side.

### *E. Communication Complexity:*

For a security parameter $\lambda$ . On the sender's side, a one bit encryption with SNCE requires the transmission of only $\lambda$ bits. On the receiver's side, RNCE still requires only $\lambda$ bits of transmission (in addition to one bit in the second pass). In a one attempt of BET scheme, a transmission of $2\lambda$ bits is required in the first pass, $4\lambda$ bits in the second pass in addition to 2 bits in the third pass. Hence a total of $6\lambda+2$ bits are required on the sender's and on the receiver's side.

## VII. REFERENCES

[1] A. C. Yao, Protocols for secure computations (extended abstract). In 23rd Annual Symposium on Foundations of Computer Science, pp 160-164.

[2] A.Herzberg, S.Jarecki, H.Krawczyk, M.Yung, "Proactive secret sharing or: How to cope with perpetual leakage", LNCS 963, Proc. Crypto'95, Springer Verlag, 1995, pp 339-352.

[3] R. Ostrovsky, M. Yung: How to Withstand Mobile Virus Attacks, (Extended Abstract), PODC 1991, pp. 51-59.

[4] Ran Canetti, Ivan Damgard, Stefan Dziembowski, Yuval Ishai, Tal Malkin: On Adaptive vs. Non-adaptive Security of Multiparty Protocols. EUROCRYPT 2001, pp 262-279.

[5] Ran Canetti, Uriel Feige, Oded Goldreich, Moni Naor, Adaptively Secure Multi-Party Computation. STOC 1996, pp. 639-648.

[6] Ran Canetti, Cynthia Dwork, Moni Naor, Rafail Ostrovsky: Deniable Encryption. CRYPTO 1997, pp. 90-104

[7] Ran Canetti, Rosario Gennaro: Incoercible Multiparty Computation (extended abstract). FOCS 1996, pp. 504-513.

[8] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. Journal of the ACM, 40(1):17-47, Jan. 1993.

[9] Kannan Srinathan, Arpita Patra, Ashish Choudhary, C. Pandu Rangan: Probabilistic Perfectly Reliable and Secure Message Transmission - Possibility, Feasibility and Optimality. INDOCRYPT 2007, pp. 101-122

[10] Arpita Patra, Ashish Choudhary, K. Srinathan, C. Pandu Rangan: Constant Phase Bit Optimal Protocols for Perfectly Reliable and Secure Message Transmission. INDOCRYPT 2006: pp. 221-235.

[11] Matthias Fitzi, Matthew K. Franklin, Juan A. Garay, S. Harsha Vardhan: Towards Optimal and Efficient Perfectly Secure Message Transmission. TCC 2007: 311-322.

[12] K. Kurosawa, K. Suzuki, Truly Efficient 2-Round Perfectly Secure Message Transmission Scheme, EUROCRYPT 2008: 324-340.

[13] E. Berlekamp and L. Welch. Error correction of algebraic block codes. US Patent 4,633,470.

[14] S. Goldwasser, S. Micali, C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems." SIAM J. on Computing 18:1, 1989, 186-208.

[15] O. Goldreich, S. Micali, A. Wigderson. "Proofs that Yield Nothing but Their Validity and a Methodology of Cryptographic Protocol Design." Proceedings of the 27 th FOCS, IEEE, 1986, 174-187.

[16] D. Beaver. "Foundations of Secure Interactive Computing." Advances in Cryptology- Crypto '91 Proceedings, Springer-Verlag LNCS 576, 1992, 377-391.

[17] D. Beaver. "Adaptive Zero Knowledge and Computational Equivocation." Proceedings of the 28 th STOC, ACM, 1996, 629-638.

[18] S. Micali, P. Rogaway. "Secure Computation." Advances in Cryptology-Crypto '91 Proceedings, Springer-Verlag LNCS 576, 1992, 392-404.

[19] D.Beaver, Foundations of secure interactive computing. In Joan Feigenbaum, editor, Advances in Cryptology - Crypto’91, pages 377--391, Berlin, 1991. Springer- Verlag. Lecture Notes in Computer Science Volume 576.

[20] Ivan Damgard and Jesoer B.Nielsen, Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, Advances in Cryptology - Crypto’2000, pages 432-450, Berlin, 2000. Springer-Verlag. Lecture Notes in Computer Science Volume 1880.

[21] J. B. Nielsen: Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case, CRYPTO 2002: 111--126.

[22] M. Bellare and P. Rogaway. Random Oracles are Practical: a Paradigm for Designing Efficient Protocols. In Proceedings of the 1st ACM Conference on Computer and Communications Security, pages 62-73.

[23] Ran Canetti, Shai Halevi, Jonathan Katz: Adaptively-Secure, Non-interactive Public-Key Encryption. TCC 2005: 150—168.

[24] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Chosen Ciphertext Attack. Crypto 1998, LNCS vol. 1462, pp. 13–25, 1998.

[25] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithm. IEEE Transactions on Information Theory, 31:465--472, 1985.

[26] P. Paillier, Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Eurocrypt 1999, LNCS vol. 1592, pp. 223–238, 1999.

[27] Pietro, R.D., Oligeri, G., Soriente, C., and Tsudik, G. Securing Mobile Unattended WSNs against a Mobile Adversary. In Proceedings of SRDS, pp. 11-20, 2010.

[28] M. H. Ibrahim, "Receiver-deniable Public-Key Encryption", Iinternational Journal of Network Security, 2009, pp.159-165.

[29] M. H. Ibrahim, "A Method for Obtaining Deniable Public-Key Encryption", International Journal of Network Security, 2009, pp.1-9.

[30] S. Lee, "Network Connection and Perfectly Secure Message Transmission on Wireless Mobile Networks", In proceedings of Teubner Studienbücher, 2010, pp.77--83.

[31] A. Patra, A. Choudhary, C. Pandu Rangan, K. Srinathan, P. Raghavendra Perfectly reliable and secure message transmission tolerating mobile adversary, International Journal of Applied Cryptography, Vol. 1, No.3 pp. 200-224, 2009.